PATENTS

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

APPLICANT(S):	Altman et al.	DOCKET: YOR920000844US1 (8728-473)
SERIAL NO.:	09/845,693	GROUP ART UNIT: 2183
FILED:	April 30, 2001	EXAMINER: Huisman, David J.
FOR:	**SYSTEM AND METHOD INCLUDING DISTRIBUTED INSTRUCTION BUFFERS HOLDING A SECOND INSTRUCTION FORM**	

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDED APPEAL BRIEF

In response to the Final Office Action dated August 8, 200, finally rejecting Claims 1, 2, 5-9, 11, 13 and 16-19 under 35 U.S.C. §102(b) and Claims 3, 10, 12, 15, 20 and 21 under 35 U.S.C. §103(a) and the Notice of Panel Decision from Pre-Appeal Brief Review dated January 10, 2007, and the Notification of Non-Compliant Appeal Brief dated May 29, 2007, Applicant appeals pursuant to the Notice of Appeal filed on November 8, 2006 and submits this appeal brief.

TABLE OF CONTENTS

1. Real Party in Interest

The real party in interest is International Business Machines Corporation, the assignee of the entire right, title, and interest in and to the subject application by virtue of an assignment of record.

2. Related Appeals and Interferences

None.

3. Status of Claims

Claims 1-3, 5-13 and 15-21 are pending, stand rejected, and are under appeal.

Claims 4 and 14 have been cancelled.

A copy of the Claims as pending is presented in the Appendix.

4. Status of Amendments

Claims 2, 5, 10-12, 15, 18, 20 and 21 were amended by the Amendment under 37 C.F.R. §1.111, filed May 13, 2004. This Amendment was entered.

Claims 1, 11, 19 and 21 were amended by the Amended under 37 C.F.R. §1.116, filed September 7, 2005. By the amendment, claims 4 and 14 were cancelled. This Amendment was entered by way of a Request for Continued Examination, filed November 8, 2004.

Claims 1, 2, 7, 11, 13 and 20 were amended by the Amendment under 37 C.F.R. §1.111, filed April 25, 2005. This Amendment was entered.

Claims 1-3 and 5-9 were amended by the Amendment under 37 C.F.R. §1.111, filed May 22, 2006. This Amendment was entered.

5. Summary of Claimed Subject Matter

A processor includes a decoder for a primary instruction form stored in the primary instruction cache or memory. The processor also includes hardware for handling an alternate form of the instruction set stored in local predecoded instruction buffers. The alternate form of the instruction is generated by a compiler.

Referring to Claim 1, a computer-implemented method for processing a first instruction set and a second instruction set in a processor comprises providing a program of instructions comprising a plurality of instructions of the first instruction set and a plurality of instructions of the second instruction set, wherein the plurality of instructions of the first instruction set are decoded by a decoder in an execution pipeline and the plurality of instructionsof the second instruction set are predecoded by a compiler (see page 6, lines 6-14). The method includes storing the plurality of instructions of the second set in a plurality of buffers proximate to a plurality of execution units (see page 9, lines 14-16 and page 10, lines 8-15), and executing at least one instruction of the first instruction set in response to a first counter (see page 11, lines 12-13). The method further includes executing at least one instruction of the second instruction set in response to at least a second counter, wherein the second counter is invoked by a branch instruction of the first instruction set, (see page 11, line 13 to page 12 line 2) wherein the step of executing at least one instruction of the second instruction set further comprises the steps of de-gating a plurality of execution queues storing the plurality of instructions of the first instruction set (see page 12, lines 3-5), and pausing a fetching of the first instruction set from a memory (see page 12, lines 5-6).

Referring to Claim 11, a processor for processing a program of instructions comprising

instructions of a first instruction form and a second instruction form comprises a plurality of execution units for receiving instructions (see Figure 3, elements 301-304 and page 10, lines 8-9), and a branch unit connected to an instruction fetch unit for the first instruction form (see Figure 3, element 305 and page 11, line 12) and a sequencer for the second instruction form, wherein the sequencer controls a plurality of gates connected between a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units (see Figure 3, element 325 and page 11, line 18 to page 12 line 11). The processor includes a decode unit for decoding instructions of the first instruction form into control signals for the execution units (see Figure 3, element 323 and page 18, line 22 to page 19, line 1), and a plurality of buffers, proximate to the execution units, for storing predecoded instructions of the second instruction form (see Figure 3, elements 306-310 and page 9, line 18-19).

Referring now to Claim 21, a processor for processing a first instruction form and a second instruction form of an instruction set comprises a plurality of execution units for receiving instructions (see Figure 3, elements 301-304 and page 10, lines 8-9), and a branch unit connected to an instruction fetch unit for the first instruction form (see Figure 3, element 305 and page 11, line 12) and a sequencer for the second instruction form, wherein the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form and switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form (see Figure 3, element 325 and page 11, line 8 to page 12, line 11). The processor comprises a decode unit adapted to decode instructions of the first instruction form into control signals for the execution units (see Figure 3, element 323 and page 18, line 22 to page 19, line 1), and an issue unit adapted to sequence decoded instructions of the first

instruction form (see Figure 3, element 324 and page 11, line 21 to page 12 line 1). The processor includes a plurality of buffers, proximate to the execution units, for statically storing predecoded instructions of the second instruction form, wherein each execution unit is connected to a corresponding buffer of the plurality of buffers (see Figure 3, elements 306-310 and page9, lines 18-19), and the sequencer, engaged by the branch unit, adapted to fetch the predecoded instructions and sequence the predecoded instructions of the second instruction form, wherein the sequencer is connected to a plurality of gates connected between a plurality of execution queues adapted to store the decoded instructions of the first instruction form and the plurality of execution units, the sequencer further adapted to control the gates (see Figure 3, element 325 and page 11, line 18 to page 12, line 11).

6. Grounds of Rejection to be Reviewed on Appeal

A. Claims 1-11, 13 and 15-20 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Lavi et al. (USPN 6,453,407) in view of Ito et al. (USPN 5,742,782).

7. Argument

A. The Claim Rejections Under 35 U.S.C. 103 Are Legally Deficient.

In rejecting claims under 35 U.S.C. §103, the Examiner bears the initial burden of presenting a *prima facie* case of obviousness. In re Rijckaert, 9 F.3d 1531, 1532 (Fed. Cir. 1993). The burden of presenting a *prima facie* case of obviousness is only satisfied by showing some objective teaching in the prior art or that knowledge generally available to one of ordinary skill in the art would lead that individual to combine the relevant teachings of the references. In re Fine, 837 F.2d 1071, 1074 (Fed. Cir. 1988). A *prima facie* case of obviousness is established when the teachings of the prior art itself would appear to have suggested the claimed subject matter to one of ordinary skill in the art. In re Bell, 991 F.2d 781, 782 (Fed. Cir. 1993). If the Examiner fails to establish a *prima facie* case, the rejection is improper and must be overturned. In re Rijckaert, 9 F.3d at 1532 (citing In re Fine, 837 F.2d at 1074).

It is respectfully submitted that at the very least, the combined teachings of Lavi and Ito are legally deficient to establish a *prima facie* case of obviousness against independent Claims 1, 11 and 21.

i. Claim 1

Claim 1 claims, *inter alia*, "providing a program of instructions comprising a plurality of instructions of the first instruction set and a plurality of instructions of the second instruction set, wherein the plurality of instructions of the first instruction set are decoded by a decoder in an execution pipeline and the plurality of instructions of the second instruction set are predecoded by a compiler."

The Examiner has indicated that "first and second instruction sets are broadly interpreted as first and second sets of instructions. Any group of instructions may be considered a set of instructions." Respectfully, such an interpretation is contrary to the plain meaning of the term "instruction set" as used in the art and further as defined in the specification. For example, consider the dictionary definition of "instruction set" given as "the repertoire of arithmetic and logical operations of a computer. All programs exist in a computer as sequences of instructions drawn from the instruction set" (see Penguin Dictionary of Electronics, 3rd Ed. 1998). Indeed, the dictionary definition cited by the Examiner in the Advisory Action supports this distinction – that is an "'instruction set' as the set of machine instructions that a processor recognizes and can execute" (emphasis added). The definition does not state that any set is an instruction set, but rather that the set of machine instructions that a processor recognizes and can execute – that is all instructions a processor recognizes and can execute. Thus, an "instruction set" is not merely a group of instructions, but the repertoire of instructions available. Given the plain meaning of "instruction set," consider the following:

Lavi teaches a Configurable Long Instruction Word (CLIW) reference instruction is a decoded Very Long Instruction Word (VLIW) instruction word (see col. 9, lines 57-61). Lavi does not teach "providing a program of instructions comprising a plurality of instructions of the first instruction set and a plurality of instructions of the second instruction set, wherein the plurality of instructions of the first instruction set are decoded by a decoder in an execution pipeline and the plurality of instructions of the second instruction set are predecoded by a compiler" as claimed in Claim 1. Lavi teaches a single instruction set stored as decoded and undecoded instructions. Decoded and undecoded instructions of the same instruction set are not analogous to a first instruction set and a second instruction set, essentially as claimed in Claim 1.

For example, Lavi's single instruction set is stored as both regular instructions and CLIW-reference instructions (see col. 16, lines 9-12); nowhere Lavi does not teach or suggest instructions and decoded instructions are from different instruction sets. Thus, Lavi fails to teach or suggest all the limitations of Claim 1.

Ito teaches that an instruction scheduler includes an instruction buffer (see element 37, Figure 3). Ito merely teaches decoding instructions. Nowhere does Ito teach or suggest an instruction set much less, a first instruction set and a second instruction set, essentially as claimed in Claim 1. Therefore, Ito fails to cure the deficiencies of Lavi.

The combined teachings of Lavi and Ito teach decoding instructions of a single instruction set. The combined teachings of Lavi and Ito fail to teach or suggest all the limitations of Claim 1; in particular "providing a program of instructions comprising a plurality of instructions of the first instruction set and a plurality of instructions of the second instruction set."

ii. Claim 11

Claim 11 claims, *inter alia*, "a branch unit connected to an instruction fetch unit for the first instruction form and a sequencer for the second instruction form, wherein the sequencer controls a plurality of gates connected between a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units."

Lavi teaches a CLIW reference instruction is discriminated from a regular instruction by an instruction decoder (see col. 10, lines 1-13). Lavi does not teach "a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units" as claimed in Claim 11. Lavi does not teach or suggest an execution queue. Lavi teaches

that instruction decoders (see element 40 of FIG. 3) operate in parallel with the CLIW array (see element 70 of FIG. 3) to further analyze all additional information stored in the reference instruction (see col. 10, lines 13-17). Nowhere does Lavi teach or suggest de-gating one of the instruction decoders or CLIW, much less a plurality of execution queues storing the plurality of instructions of the first instruction form. Indeed, from FIG. 3 it is clear that data is passed directly to an execution layer – there is no execution queue. Accordingly, Lavi fails to teach or suggest all the limitations of Claim 11.

Ito teaches that an instruction scheduler includes an instruction buffer (see element 37, Figure 3). Ito does not teach "a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units" as claimed in Claim 11. Ito teaches that an instruction buffer stores instructions prior to decoding. Thus, at the very least, the instruction buffer of Ito is not an execution queue – the instructions stored in Ito's instruction buffers cannot be executed because they have not yet been decoded, thus the instruction buffer is not an execution queue. One of ordinary skill in the art would understand that an execution queue stores decoded instructions. Further, it is clear that the instruction buffer is not de-gated as only one instruction buffer is provided. De-gating the instruction buffer of Ito would terminate an executing program. Therefore, Ito fails to cure the deficiencies of Lavi.

The combined teachings of Lavi and Ito teach decoding instructions of a single instruction set. The combined teachings of Lavi and Ito fail to teach or suggest all the limitations of Claim 11; in particular "a plurality of gates connected between a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units."

iii. Claim 21

Claim 21 claims "the sequencer, engaged by the branch unit, adapted to fetch the predecoded instructions and sequence the predecoded instructions of the second instruction form, wherein the sequencer is connected to a plurality of gates connected between a plurality of execution queues adapted to store the decoded instructions of the first instruction form and the plurality of execution units, the sequencer further adapted to control the gates.

Lavi teaches a CLIW reference instruction is discriminated from a regular instruction by an instruction decoder (see col. 10, lines 1-13). Lavi does not teach or suggest "a branch unit connected to an instruction fetch unit for the first instruction form and a sequencer for the second instruction form, wherein the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form and switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form" as claimed in Claim 21. Lavi teaches branch instructions for proceeding to a different location in a program (see col. 3, lines 30-34) – Lavi does not teach or suggest a branch instruction for switching between operation of the instruction decoder and the CLIW array – as shown with respect to Claim 11 the instruction decoder and the CLIW array of Lavi operate in parallel. Indeed, the program of Lavi includes both regular instructions and reference instructions in order (see Figure 6) - no branch instruction is needed to switch between regular and reference instructions. The branch instruction operates to change a location without any knowledge of regular and reference instructions. Further, the instructions of the instruction decoder and CLIW array are not of different forms – they are merely the result of different instructions of the same instruction set fetched from the program memory. The normal and reference instructions are fetched along the same pathway; Lavi has no

need for a branch instruction back to the normal instructions because they are fetched along the same pathway as the reference instructions. Thus, Lavi fails to teach or suggest all the limitations of Claim 21.

Ito teaches that an instruction scheduler includes an instruction buffer (see element 37, Figure 3). Nowhere does Ito teach or suggest a branch instruction, much less switching a processor from a first instruction form to a second instruction form in response to a branch instruction, essentially as claimed in Claim 21. Indeed, Ito fails to teach or suggest a first and a second instruction form. Thus, Ito fails to cure the deficiencies of Lavi.

The combined teachings of Lavi and Ito teach decoding instructions of a single instruction set. The combined teachings of Lavi and Ito fail to teach or suggest all the limitations of Claim 21; in particular a "branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form and switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form."

B. CONCLUSION

The claimed invention is not disclosed or suggested by the teachings of the applied prior art references, either alone or in combination. Moreover, the Examiner has failed to establish a *prima facie* case of obviousness under 35 U.S.C. §103 over Lavi and Ito with respect to Claims 1, 11 and 21 for at least the reasons noted above. Claims 2, 3 and 5-10 depend from Claim 1. Claims 12, 13 and 15-20 depend from Claim 11. The dependent claims are believed to be allowable for at least the reasons given for Claims 1 and 11. Accordingly, it is respectfully requested that the Board overrule the rejections of Claims 1-3, 5-13 and 15-21.

Respectfully submitted,

Date: June 29, 2007

By: /Nathaniel T. Wallace/
Nathaniel T. Wallace
Reg. No. 48,909
Attorney for Appellants

F. CHAU & ASSOCIATES, LLP
130 Woodbury Road
Woodbury, New York 11797
TEL: (516) 692-8888
FAX: (516) 692-8889

8. CLAIMS APPENDIX

1. A computer-implemented method for processing a first instruction set and a second instruction set in a processor comprising the steps of:

providing a program of instructions comprising a plurality of instructions of the first instruction set and a plurality of instructions of the second instruction set, wherein the plurality of instructions of the first instruction set are decoded by a decoder in an execution pipeline and the plurality of instructions of the second instruction set are predecoded by a compiler;

storing the plurality of instructions of the second set in a plurality of buffers proximate to a plurality of execution units;

executing at least one instruction of the first instruction set in response to a first counter; and

executing at least one instruction of the second instruction set in response to at least a second counter, wherein the second counter is invoked by a branch instruction of the first instruction set,

wherein the step of executing at least one instruction of the second instruction set further comprises the steps of

de-gating a plurality of execution queues storing the plurality of instructions of the first instruction set, and

pausing a fetching of the first instruction set from a memory.

2. The method of claim 1, wherein the instructions of the first set and instructions of the second set are generated by a compiler, wherein instructions of the second set are statically loaded into the plurality of buffers as control signals ready for execution.

3. The method of claim 2, wherein instructions of the second set are more frequently executed than instructions of the first set.

5. The method of claim 1, wherein the step of executing at least one instruction of the second instruction set further comprises the steps of:

fetching at least one instruction of the second instruction set from a buffer of the plurality of buffers; and

sequencing the at least one instruction of the second instruction set to the execution units.

6. The method of claim 1, wherein the second instruction set is a logical subset of the first instruction set.

7. The method of claim 1, wherein the step of executing at least one instruction of the first instruction set further comprises the steps of:

fetching an instruction of the first form set a memory;

decoding the instruction; and

issuing the decoded instruction to at least one execution unit.

8. The method of claim 1, wherein a return to fetching of the first instruction set is signaled by a switch bit in a buffer of a branch unit storing instructions of the second set.

9. The method of claim 1, wherein a return to fetching of the first instruction set is signaled

by a return instruction of the second instruction set stored in a buffer of a branch unit.

10. The method of claim 1, wherein each execution unit is associated with a different buffer of the plurality of buffers.

11. A processor for processing a program of instructions comprising instructions of a first instruction form and a second instruction form comprising:

a plurality of execution units for receiving instructions;

a branch unit connected to an instruction fetch unit for the first instruction form and a sequencer for the second instruction form, wherein the sequencer controls a plurality of gates connected between a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units;

a decode unit for decoding instructions of the first instruction form into control signals for the execution units; and

a plurality of buffers, proximate to the execution units, for storing predecoded instructions of the second instruction form.

12. The processor of claim 11, wherein the instructions of the first form and instructions of the second form are generated based on execution frequency, wherein instructions of the second form are executed more frequently than instructions of the first form.

13. The processor of claim 11, wherein the sequencer, engaged by the branch unit, addresses the predecoded instructions of the second instruction form stored in the buffers and sequences

predecoded instructions of the second instruction form to the execution unit.

15. The processor of claim 11, wherein each execution unit is connected to a corresponding buffer of the plurality of buffers.

16. The processor of claim 11, wherein the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form.

17. The processor of claim 11, wherein the branch unit switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form.

18. The processor of claim 11, wherein a switch bit in a buffer of the plurality of buffers connected to the branch unit signals the sequencer to stop fetching from the buffers and enables instruction fetching from a memory storing instructions of the first instruction form.

19. The processor of claim 11, wherein an execution bandwidth of the execution units is larger than a fetch/issue bandwidth of the first form.

20. The processor of claim 11, wherein the second instruction form is a logical subset of the first instruction form, wherein the predecoded instructions of the second instruction form are statically stored in the plurality of buffers, and wherein the predecoded instructions of the second

instruction form are control signals generated by a compiler and are not decoded during a runtime of the program.

21. A processor for processing a first instruction form and a second instruction form of an instruction set comprising:

a plurality of execution units for receiving instructions;

a branch unit connected to an instruction fetch unit for the first instruction form and a sequencer for the second instruction form, wherein the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form and switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form;

a decode unit adapted to decode instructions of the first instruction form into control signals for the execution units;

an issue unit adapted to sequence decoded instructions of the first instruction form;

a plurality of buffers, proximate to the execution units, for statically storing predecoded instructions of the second instruction form, wherein each execution unit is connected to a corresponding buffer of the plurality of buffers; and

the sequencer, engaged by the branch unit, adapted to fetch the predecoded instructions and sequence the predecoded instructions of the second instruction form, wherein the sequencer is connected to a plurality of gates connected between a plurality of execution queues adapted to store the decoded instructions of the first instruction form and the plurality of execution units, the sequencer further adapted to control the gates.

9. EVIDENCE APPENDIX

NONE

10. RELATED PROCEEDINGS APPENDIX

NONE